REMARKS

Claims 1, 3-22, 24-43 and 45-53 are pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Section 103(a) Rejections:

The Examiner rejected claims 1, 6-12, 16-22, 27-34, 40-43, 45-49 and 51-53 under 35 U.S.C. § 103(a) as being unpatentable over Erickson et al. (U.S. Patent 6,851,089) (hereinafter "Erickson") in view of Germscheid et al. (U.S. Patent 6,782,425) (hereinafter "Germscheid"), and claims 3-5, 13-15, 24-26, 35-39 and 50 as being unpatentable over Erickson and Germscheid in view of Wu (U.S. Patent 5,774,551). Applicants respectfully traverse this rejection for at least the reasons below.

Regarding claim 1, Erickson in view of Germscheid fails to teach or suggest the client device receiving a message in the data representation language from a service device in the distributed computing environment prior to generating a computer programming language object, wherein the message includes the data representation language representation of the object. The Examiner cites column 26, lines 15-20 of Erickson. However, the Examiner's cited passage describes that a wrapper builder application uses serialization to encode an internal object representation of a wrapper into XML format. This cited passage does not mention anything about a *message* including a data representation language representation of a computer programming language object, let alone a client device receiving such a message from a service device in a distributed computing environment.

Erickson teaches that his wrapper builder and wrapper execution applications load wrappers using serialization *locally from disk*. For example, figures 14 and 15 both illustrate a wrapper file *on the same computer as the wrapper builder or wrapper execution applications*. Erickson teaches that after a "wrapper file has been created and stored, the wrapper file can be read by a wrapper builder application and deserialized"

(italics added, Erickson, column 26, lines 21 – 29). Thus, Erickson teaches that a wrapper file is *read*, such as from a local disk file as illustrated in figures 14 and 15. The wrapper file is clearly not a *message that a client device receives from a service device in a distributed computing environment*. Erickson does not mention anything about *a client device receiving a message* that includes a data representation language of a computer programming language object *from a service device in a distributed computing environment*. **Reading a serialized wrapper file locally is quite different from receiving a message from another device that includes a data representation language representation of a computer programming language object.**

In the Response to Arguments section of the Final Action, the Examiner states, "Erickson does not specify reading the wrapper file locally" and "Erickson clearly discloses a wrapper serialization component that provides for the storage and retrieval of wrappers in XML through the process of Object Serialization and serialization is used for communications via sockets or remote method invocation", citing column 25, line 56 – column 26, line 15 of Erickson. Firstly, as stated above, Erickson repeatedly states that wrapper files may be *stored* and *read* from disk. Additionally, as noted above, Erickson's figures 14 and 15 illustrate that the wrappers files are stored and read *locally*. **Thus, contrary to the Examiner statement, Erickson clearly teaches that wrapper files are stored to and read from the local machine.** The Examiner is relying upon Erickson's quoting of an excerpt from a Sun Microsystems Java web page that summarizes the *concept* of serialization. While Erickson, by including this excerpt summarizing the *concept* of serialization, mentions that serialization, in general, may be used for communication via sockets or RMI, Erickson's actual system does not use serialization for communication via sockets or RMI. Instead, as shown above Erickson uses serialization to store wrappers in wrapper files, which can subsequently be read from the wrapper file, illustrated in figures 14 and 15 as local to the same machine, "for subsequent execution or editing through the process of serialization."

The Examiner has not cited any portion of Erickson, whether considered singly or in combination with Germscheid, that describes, teaches, or even implies that Erickson's

wrappers or wrapper files are sent in messages that include a data representation language representation of a programming language object from a service in a distributed computing environment. The Examiner relies solely on the general description of serialization at column 25, line 56 – column 26, line 15 of Erickson. Indeed, the Examiner is relying upon a single sentence stating that serialization may be used for "communications via sockets or Remote Method Invocation (RMI)." Erickson makes no other mention of either communication via sockets or RMI.

The Examiner also states in the Response to Arguments that Erickson's "wrapper files [message] stored as an XML is retrieved from wrapper serialization component [service device] via sockets or remote method invocation [distributed computing environment] (brackets by Examiner), again citing column 25, line 56 – column 26, line 15 of Erickson. Applicants respectfully disagree with the Examiner's interpretation of Erickson. Nowhere does Erickson, whether considered singly or in combination with Germscheid, teach that his wrapper files are sent or retrieved via sockets or RMI. The fact that serialization may be used, *in general*, "for communication via sockets and Remote Method Invocation (RMI)" does not have any bearing on the fact that Erickson in view of Germscheid fails to teach or suggest a client device receiving a wrapper file, which the Examiner equates to Applicants' message, from a service device in the distributed computer environment. In contrast, in his general teachings regarding serialization, Erickson specifically teaches that in his system, the "wrapper builder employs serialization to encode an internal object representation of a wrapper into XML format" that "can be saved as a wrapper file". **Furthermore, the Examiner not provided any argument or explanation regarding how a single, general reference to "communications via sockets and Remote Method Invocation (RMI)" can be interpreted as teaching the specific functionality of sending Erickson's wrapper files to a client machine when Erickson makes no mention of sending wrapper files, receiving wrapper files, but specifically discloses storing wrapper files to and reading them from a local drive.**

Erickson further teaches, "once a wrapper file has been created and stored, the wrapper file can be read by a wrapper builder application and deserialized" (Erickson, column 26, lines 16-26). Even when deployed for use on a host system connected to the Internet, Erickson specifically teaches that a wrapper execution engine reads and deserializes wrapper files that are stored locally (see, FIG. 15 and column 27, lines 9-20 and 41-59). Erickson is not concerned with communicating wrappers or wrapper files over sockets or RMI. Instead, Erickson is concerned with creating and saving wrappers that can subsequently be *read* from disk for "execution or editing." **Erickson does not teach that serialization is used to send the wrapper file as a message (via RMI or any other technique) from a service device to a client device. To the contrary, Erickson clearly and explicitly states that serialization is used only to encode an internal object representation of a wrapper into XML which is then stored *on the same computer* for later retrieval and use *on the same computer*. The wrapper file is not sent as a message from a service device to a client device.**

Please note that the above arguments refer to Erickson to demonstrate that the Examiner's reliance on Erickson is misplaced. As will be shown below, Applicants argue that the combination of Erickson in view of Germscheid fails to teach or suggest all the limitations of Applicants' claim.

Germscheid also fails to teach or suggest anything regarding a client device receiving a message in a data representation language from a service device in the distributed computing environment, wherein the message includes a data representation language representation of a computer programming language object. Germscheid therefore fails to overcome the above-noted deficiencies of Erickson with regard to teaching or suggesting a client device receiving a message in a data representation language from a service device, where the message includes a data representation language representation of a computer programming language object. Thus, the Examiner's combination of Erickson and Germscheid would not include such functionality. Instead, a combination of Erickson and Germscheid would result in a

system that loaded wrapper files, via serialization, locally from disk, as taught by Erickson.

Additionally, the Examiner has failed to provide a proper motivation to combine Erickson and Germscheid. The Examiner gives two reasons why one would be motivated to combine Germscheid's teachings regarding deleting a session object when a user terminates a session with Erickson's system. Firstly, the Examiner states that it would be obvious to combine Erickson and Germscheid "because this would prevent unauthorized access to the object." However, Erickson is not concerned with, nor does Erickson mention anything about, security or preventing unauthorized access to objects. The fact that Germscheid teaches deleting an object to prevent unauthorized access does not have any bearing on the teachings of Erickson or provide any motivation to modify Erickson's teachings. Since Erickson is not concerned at all with security or preventing unauthorized access, Germscheid does not provide any motivation to modify Erickson's system. **In fact, Erickson teaches that his wrapper serialization component should persist for subsequent use (see, e.g., Abstract, last sentence). Thus, Erickson actually teaches away from the Examiner's proposed combination.**

In the Response to Arguments, the Examiner states, "a person of ordinary skill in the art would know that deleting an object will remove the object from the shared memory of the computer and prevent unauthorized assess (sic) to the object." However, Applicants have not argued that deleting an object would not remove the object from shared memory of a computer or prevent unauthorized access. Instead, as discussed above, Applicants argue that since Erickson is not concerned with or mention anything about security or preventing unauthorized access to objects, no one would be motivated to modify Erickson to include the object deletion taught by Germscheid. Moreover, as noted above, Erickson **teaches away** from a combination with Germscheid.

The Examiner has previously argued that "[s]ince Erickson discloses desire to protect private and transient data of an object, it would have been obvious to combine Erickson's invention with Germscheid in order to prevent unauthorized access to the

object." The Examiner again relies on Erickson's discussion of Java serialization, citing column 26, lines 10-15 of Erickson. Specifically, the Examiner contends that Erickson discloses "security and preventing unauthorized access by encoding objects to protect private and transient data." However, at the cited passage, Erickson is describing Java serialization in general, not his actual invention. Moreover, the cited passage teaches that "the default encoding of objects [in Java serialization] protects private and transient data." The Examiner is clearly incorrect to state that Erickson "discloses a desire to protect private and transient data of an object". A general statement that serialization protects private and transient data does not override the fact that Erickson's system is concerned with making the wrapper objects available for subsequent use to access web sites. Erickson's system is specifically directed toward making wrapper objects available for use to access web sites. Erickson's general statement that Java serialization "protects private and transient data" does not provide motivation to combine Erickson with Germscheid. The Examiner's reliance on a single, general statement regarding Java serialization to provide motivation to combine Erickson and Germscheid to teach the specific combination of features recited in Applicants' claim is clearly misplaced.

The Examiner also states that it would be obvious to modify Erickson according to the teachings of Germscheid because deleting the object "deallocates the storage for the object after the user has finished accessing the object." This motivation is not supported by any evidence of record and seems to be the Examiner's own opinion formed in hindsight. Neither Germscheid nor Erickson, whether considered singly or in combination, mentions anything about deallocating storage. Moreover, as noted above, Erickson teaches that his wrapper should persist for subsequent use (see, e.g., Abstract, last sentence). Thus, Erickson actually teaches away from the Examiner's proposed combination.

In the Response to Argument, the Examiner asserts, "one of ordinary skill in the art would now to delete an object and reclaim the memory for the object" and that in addition, "Erickson discloses the Java environment [col. 3, lines 25 – 36] and garbage collection is a feature of Java." Thus, the Examiner contends that since Erickson's

system may be implemented in Java, and since Java includes garbage collection, it would be obvious to modify Erickson to include the specific functionality taught by Germscheid regarding deleting a CCISession object after the termination of Cool ICE session. Firstly, Erickson makes no mention of using or relying on the garbage collection feature of Java. Secondly, just because Erickson's system may be implemented in Java does not provide any motivation to include the CCISession object deletion of Germscheid, much less extend Germscheid's teaching to include deleting Erickson's wrapper object. Germscheid does not teach object deletion in response to a user terminating accessing a client device. Instead, Germscheid teaches specifically deleting a CCISession object after termination of a Cool ICE session. Thirdly, as stated by the Examiner, Germscheid teaches deleting the CCISession object to prevent unauthorized access. However, as noted above, Erickson is not concerned with security or unauthorized access. The motivations stated by the Examiner apply only to Germscheid's system and have no bearing on Erickson's. Thus the Examiner's stated motivation would not motivate one to include the CCISession object deletion of Germscheid in Erickson's system. Also, as noted above, Erickson actually teaches away from the Examiner's proposed modification.

The Examiner has also asserted that in Erickson system, "[w]hen the wrapper file is read from storage a copy of the wrapper file would be created" citing column 26, lines 20-30, and argues, "[i]t would be obvious to delete the copy of the wrapper file when the client is finished with the temporary copy of the wrapper file." However, the Examiner's interpretation of Erickson is incorrect. Firstly the Examiner relies on the phrase "reproduce the wrapper" when erroneously concluding that Erickson's system makes a copy of a wrapper file. However, the cited passage does not describe making a temporary copy of a wrapper file, as the Examiner suggests. Instead, Erickson states, "a wrapper execution engine … can read the serialized wrapper (e.g. from the wrapper file), reproduce the wrapper within its execution environment, and run it (the wrapper)" (parenthesis added). Thus, Erickson is describing that after a wrapper file has been created and stored, a wrapper execution engine can read the wrapper file, deserialize the wrapper itself (i.e. reproduce the wrapper within its execution environment) and execute the wrapper. Erickson is clearly not discussing creating a temporary copy of a wrapper

file, as the Examiner suggests. Secondly, and more importantly, even if Erickson were to teach deleting a temporary copy of a wrapper file, as the Examiner erroneously contends, **that would not change the fact that Erickson's entire system is directed toward the design and creation of wrapper files for subsequent use when accessing web sites** (Erickson, FIG. 15; column 2, lines 58-65; and column 27, lines 9-59) and not preventing unauthorized access, the Examiner stated motivation for combination Erickson and Germscheid. **Thus, Erickson clearly teaches away from the Examiner proposed combination.**

Furthermore, even if the combination was properly motivated (which it is not), the Examiner's combination of Erickson and Germscheid fails to teach deleting, in response to a user terminating accessing a client device, a computer programming language object that was generated from a data representation language representation of the object. Erickson teaches deserializing wrapper files to generate wrapper objects used to access Internet content. Germscheid teaches deleting security session objects as part of terminating a user's session. Thus, when combined as suggested by the Examiner, the resultant system would only include the wrapper deserialization and Internet content access as taught by Erickson and deleting any security session objects, as taught by Germscheid. Erickson's wrapper components are not security session objects. Thus, while the combination may result in deleting security session objects in Erickson's system employing wrapper components, it would not suggest deleting the wrapper components themselves. In fact, as noted above, Erickson actually teaches away from deleting the wrapper components to not be accessible for use by subsequent users.

In the Response to Arguments, the Examiner contends, "the security session objects of Germscheid corresponds to the wrapper objects of Erickson because both objects provides access to a network resource." However, the Examiner's interpretation is incorrect. Germscheid's security session objects and Erickson's wrapper objects are two very different types of object that perform two very different functions. Germscheid teaches that his security session objects store user login information (Germscheid,

column 4, lines 54-63). In contrast, Erickson teaches that wrappers are "capable of the automated extraction of data from Internet or intranet Web sites" (Erickson, column 58-65). The Examiner's over generalized statement that they both provide access to a network resource does not overcome the fact that Erickson's wrapper components are not security session objects as taught by Germscheid. Thus, the Examiner's combination may result in deleting security *session objects* in Erickson's system, but it would not suggest deleting the wrapper components themselves.

Thus, for at least the reasons above, the rejection of claim 1 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks apply to claims 22 and 43.

Regarding claim 10, Erickson in view of Germscheid fails to teach or suggest a client device receiving a message in a data representation language from a service device in the distributed computing environment, wherein the message includes a data representation language representation of a computer programming language object. The Examiner cites column 16, lines 1-30 and column 26, lines 15-20 of Erickson. However, the first cited passage (column 16, lines 1-30) describes an "If" operation that is part of Erickson's system. As noted above regarding the rejection of claim 1, nothing about Erickson's "If" operation or the cited passage has anything to do with a client device receiving a message in a data representation language from a service device in the distributed computing environment. The Examiner's other cited passage (column 26, lines 15-20) describes that a wrapper builder application uses serialization to encode an internal object representation of a wrapper into XML format. However, this cited passage does not mention anything about a *message* including a data representation language representation of a computer programming language object, let alone a client device receiving such a message from a service device in a distributed computing environment. Erickson teaches that his wrapper builder and wrapper execution applications load wrappers using serialization *locally from disk*. For example, figures 14 and 15 both illustrate a wrapper file on the same computer as the wrapper builder or wrapper execution applications. Erickson teaches that after a "wrapper file has been

created and stored, the wrapper file can be read by a wrapper builder application and deserialized" (italics added, Erickson, column 26, lines 21 – 9). Thus, Erickson teaches that a wrapper file is *read*, such as from a local disk file as illustrated in figures 14 and 15. Erickson does not mention anything about a client device receiving a message that includes a data representation language of a computer programming language object from a service device in a distributed computing environment. Reading a serialized wrapper file locally is quite different from receiving a message from another device that includes a data representation language representation of a computer programming language object.

In the Response to Arguments, the Examiner states, "Erickson does not specify reading the wrapper file locally" and "Erickson clearly discloses a wrapper serialization component that provides for the storage and retrieval of wrappers in XML through the process of Object Serialization and serialization is used for communications via sockets or remote method invocation", citing column 25, line 56 – column 26, line 15 of Erickson. As stated above, Erickson repeatedly states that wrapper files may be *stored* and *read* from disk. Additionally, as noted above, Erickson's figures 14 and 15 illustrate that the wrappers files are stored and read *locally*. Thus, contrary to the Examiner statement, Erickson clearly teaches that wrapper files are stored to and read from the local machine. The Examiner is relying upon Erickson's quoting of an excerpt from a Sun Microsystems Java web page that summarizes the *concept* of serialization. While Erickson, by including this excerpt summarizing the *concept* of serialization, mentions that serialization, in general, may be used for communication via sockets or RMI, Erickson's system does not use serialization for communication via sockets or RMI. Instead, as argued above Erickson uses serialization to store wrappers in wrapper files, which can subsequently be read from the wrapper file, illustrated in figures 14 and 15 as local to the same machine, "for subsequent execution or editing through the process of serialization."

The Examiner has not cited any portion of Erickson, whether considered singly or in combination with Germscheid, that describes, teaches, or even implies that Erickson's wrappers or wrapper files are send in messages that include a data representation

language representation of a programming language object from a service in a distributed computing environment, as suggested by the Examiner. The Examiner relies solely on the general description of serialization at column 25, line 56 – column 26, line 15 of Erickson. Indeed, the Examiner is relying upon a single sentence stating that serialization may be used for "communications via sockets or Remote Method Invocation (RMI)." Erickson makes no other mention of either communication via sockets or RMI.

Additionally, Germscheid, whether considered singly or in combination with Erickson, fails to teach or suggest anything regarding a client device receiving a message in a data representation language from a service device in the distributed computing environment, wherein the message includes a data representation language representation of a computer programming language object. Germscheid fails to overcome the above-noted deficiencies of Erickson with regard to teaching or suggesting a client device receiving a message in a data representation language from a service device, where the message includes a data representation language representation of a computer programming language object. Thus, the Examiner's combination of Erickson and Germscheid would not include such functionality. Instead, a combination of Erickson, Germscheid and Wu, would result in a system that loaded wrapper files, via serialization, locally from disk, as taught by Erickson.

Additionally, the Examiner's combination of Erickson and Germscheid does not result in a system that teaches or suggests generating a computer programming language object from a data representation language representation of the object if it is determined that the user has access rights to the computer programming language object. Instead, the Examiner's combination of Erickson and Germscheid would result only in a system that includes wrapper building and executing applications as taught by Erickson, but also includes the conditional generation of security session objects as taught by Germscheid. Generating security objects if a user has proper access rights does not teach or suggest modifying Erickson's system to include *deserializing* a wrapper file if it is determined that a user has access rights to the wrapper, as the Examiner's interpretation requires.

Germscheid's teachings are in a completely different context that the wrapper deserialization described in Erickson.

In the Response to Arguments, the Examiner again argues, "the security session objects of Germscheid corresponds to the wrapper objects of Erickson because both objects provides access to a network resource." However, the Examiner's interpretation is incorrect. As argued above regarding claim 1, Germscheid's security session objects and Erickson's wrapper objects are two very different types of object that perform two very different functions. Germscheid teaches that his security session objects store user login information (Germscheid, column 4, lines 54-63). In contrast, Erickson teaches that wrappers are "capable of the automated extraction of data from Internet or intranet Web sites" (Erickson, column 58-65). The Examiner's over generalized statement that they both provide access to a network resource does not overcome the fact that Erickson's wrapper components are not security session objects as taught by Germscheid. Thus, the Examiner's combination may result in deleting security *session objects* in Erickson's system, but it would not suggest deleting the wrapper components themselves.

Furthermore, the Examiner has failed to provide any motivation for combining Germscheid's use of security profiles with Erickson's wrapper builder and execution applications. Germscheid is not concerned with wrappers for Internet content and Erickson is not concerned with security or the use of security profiles. Also, the Examiner's stated motivation to combine Erickson and Germscheid in the rejection of claim 1, namely that deleting a session object when a user terminates a session would prevent unauthorized access to the object and deallocates the storage for the object does not have anything to with conditionally generating a computer programming language object from a data representation language representation of the object based on determining if the user has access rights to the object. Thus, the Examiner has failed to provide any motivation to combine the Germscheid's use of security profiles with Erickson's system.

As held by the U.S. Court of Appeals for the Federal Circuit in *Ecolochem Inc. v. Southern California Edison Co.*, an obviousness claim that lacks evidence of a suggestion or motivation for one of skill in the art to combine prior art references to produce the claimed invention is defective as hindsight analysis. In addition, the showing of a suggestion, teaching, or motivation to combine prior teachings "must be clear and particular . . .. Broad conclusory statements regarding the teaching of multiple references, standing alone, are not 'evidence'." *In re Dembiczak*, 175 F.3d 994, 50 USPQ2d 1614 (Fed. Cir. 1999). The art must fairly teach or suggest to one to make the specific combination as claimed. That one achieves an improved result by making such a combination is no more than hindsight without an initial suggestion to make the combination. As noted above, the Examiner has not provided a proper motivation to one to make the specific combination as claimed for combining the teachings of Erickson and Germscheid. The Examiner provided no discussion whatsoever of a motivation to combine in regard to Applicants' claim 10. Thus, the Examiner has failed to state a *prima facie* rejection of claim 10.

Thus, for at least the reasons above, the rejection of claim 10 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks apply to claims 32 and 47.

Regarding claim 11, Erickson in view of Germscheid fails to teach or suggest wherein the message further includes access information for the computer programming language object, wherein said determining if the user has access rights to the computer programming object uses the access information. The Examiner cites column 7, lines 38-50 of Germscheid. However, this passage of Germscheid makes no mention of a message including access information. Instead, this passage describes Germscheid's "Service C" which is Germscheid's highest level of classification. Germscheid teaches various categories of services. Service C represents the highest level of classification. Germscheid teaches that some services may be so sensitive that no access to the data and functions of the service is provided over the Internet. Instead, users "must take the trouble to achieve access via an old-fashioned dedicated link." Thus, the Examiner's

cited passage makes no mention of anything regarding a message including access information for a computer programming language object. In fact, Germscheid fails to teach or suggest anything regarding including accessing information for a computer programming language object in a message that also include a data representation language representation of the computer programming language object.

In the Response to Arguments, the Examiner refers to Germscheid's service levels, arguing, "[t]he security levels as defined by security profiles regulate access to a resource" citing column 7, line 60 – column 8, line 20 of Germscheid. However, Germscheid's security profiles are not included in a message that includes a data representation language representation of a computer programming language object. Germscheid teaches that security profiles are stored on the server and used to request login information from clients to ensure that only authorized users access certain types of information (Germscheid, column 14, lines 20-67). Thus, Germscheid's security profiles are not included in messages at all. Thus, as discussed above, Germscheid, whether considered singly or in combination with Erickson, fails to teach or suggest anything regarding *including access information* for a computer programming language object *in a message* that also include a data representation language representation of the computer programming language object.

Erickson fails to overcome the deficiencies of Germscheid, as they both also fail to teach or suggest anything regarding a message including access information for a computer programming language object. Thus, the Examiner's combination of Erickson and Germscheid fails to teach or suggest wherein the message further includes access information for the computer programming language object, wherein said determining if the user has access rights to the computer programming object uses the access information.

In response, the Examiner again argues, "Erickson teaches sending the wrapper file as a message from a service device to a client", referring to Erickson's including of a general summary of serialization and citing column 25, line 56 – column 16, line 15.

However, as discussed above regarding claims 1 and 10, Erickson does not teach or describe sending a wrapper file as a message. The Examiner is relying on a single sentence from the general description of object serialization that "serialization is used … for communication via sockets or Remote Method Invocation (RMI)." As noted above, while serialization may be used for communication via sockets and RMI in other contexts, nowhere does Erickson ever mention sending wrapper files via sockets, RMI or any other communication method. Furthermore, the Examiner not provided any argument or explanation regarding how a single, general reference to "communications via sockets and Remote Method Invocation (RMI)" can be interpreted as teaching the specific functionality of sending Erickson's wrapper files to a client machine when Erickson makes no mention of sending wrapper files, receiving wrapper files, but specifically discloses storing wrapper files to and reading them from a local drive.

For at least the reasons above, the rejection of claim 11 is not supported by the prior art and removal thereof is respectfully requested. Similar remarks as those above regarding claim 11 also apply to claims 33 and 48.

Regarding claim 16, the Examiner's combination of Erickson, Germscheid and Wu fails to teach or suggest storing the computer programming language object in response to the user terminating accessing the client device. The Examiner, both in the rejection and the Response to Arguments, cites column 26, lines 21-30 of Erickson. However, this portion of Erickson only mentions storing a wrapper file, in general, after a developer has generated the wrapper file and before any use of the file. This is readily apparent from the cited passage. For instance, Erickson states, "[o]nce a wrapper has been created and stored, the wrapper file can be read by a wrapper builder application and deserialized" (Erickson, column 26, lines 21-23). Erickson fails to describe, either at the Examiner's cited passage or elsewhere, storing a wrapper *in response to a user terminating accessing* a client device.

Germscheid and Wu also fail to teach or suggest anything about storing a computer programming language object in response to a user terminating accessing a

client device and thus fail to overcome the deficiencies of Erickson in this regard. Thus, the Examiner's combination of Erickson, Germscheid and Wu would not result in a system that includes storing a computer programming language object in response to a user terminating accessing a client device.

Additionally, as relied upon by the Examiner in the rejection of claims 1, 7, 12, 22, 27, 34, 43, Germscheid teaches the deletion of objects in response to a user terminating access in order to prevent unauthorized access. Thus, **Germscheid teaches away** from storing a computer programming language object in response to a user terminating accessing a client device. Since, Germscheid teaches away from storing a computer programming language object in response to user terminating accessing a client device, the Examiner's combination of Erickson, Germscheid and Wu is improper.

In the Response to Arguments, the Examiner asserts, "the claims recites both storing and deleting a computer programming language object in response to a user terminating accessing to client device" and that "Applicant is claiming both features but are arguing them separately on the basis that one feature would teach away from the other." Applicants respectfully disagree and submit that the Examiner has misinterpreted Applicants' arguments. Applicants have argued that both Erickson and Germscheid teach away from a combination with the other for various reasons. Applicants have not argued that deleting an object in response to terminating accessing teaches away from storing an object in response to terminating accessing or that storing an object in response to terminating accessing teaches away from deleting an object in response to terminating accessing. Instead, Applicants argue that Germscheid specifically teaches away from storing a computer programming language object because Germscheid specifically teaches the deletion of objects in response to a user terminating access in order to prevent unauthorized access. In fact, Germscheid states, "[t]his deletion is **important** in further protecting secure accesses" (bolding added, Germscheid, column 16, lines 4-5). Thus, Germscheid teaches that deleting an object after terminating access is important and therefore teaches away from storing an object after terminating accessing, as recited in Applicants' claim.

For at least the reasons above, the rejection of claim 16 is not supported by the prior art and removal thereof is respectfully requested. Similar remarks apply to claims 37 and 51.

Regarding claim 18, the Examiner's combination of Erickson, Germscheid and Wu fails to teach or suggest storing access rights information of the user with the object, wherein accessing the stored object comprises verifying the access rights of a user with the stored access rights information. The Examiner cites column 7, line 60 – column 8, line 3 of Germscheid. However, the cited portion of Germscheid does not mention anything about storing access rights information of the user with the object. Instead, the cited passage describes how Germscheid's lowest level of security does not require a security profile since any member of the general public is granted access permission.

In the Response to Arguments, the Examiner again cites 7, line 60 – column 8, line 3 of Germscheid and argues that Germscheid's "service levels correspond to the recited access information because the service levels regulate access to resources and accessing the stored object comprises verifying the access rights of the user with stored access rights information." However, the Examiner fails to explain where Germscheid teaches storing access rights information *with the object*. The Examiner has apparently failed to consider the fact that nowhere does Germscheid mention storing his security profile and level information with an computer programming language object generated from a data representation language representation of the object, as recited in Applicants' claims.

Erickson and Wu both fail to teach or suggest storing access rights information of a user with the object and thus both fail to overcome Germscheid's deficiencies regarding a failure to teach storing access rights information of a user with the object. Thus, the Examiner's combination of Erickson, Germscheid and Wu would not result in a system that includes storing access rights information of the user with the object, wherein

accessing the stored object comprises verifying the access rights of a user with the stored access rights information.

Furthermore, Germscheid teaches that security profiles are stored along with the command language script corresponding to a respective service request being serviced. Germscheid teaches that the security profile, which is added to the command language script file at the time of service request development by the service request developer, is stored in a repository. (Germscheid, column 14, lines 45 – 60). Thus, **Germscheid teaches away** from storing access rights information of the user with the object, wherein accessing the stored object comprises verifying the access rights of a user with the stored access rights information. As such, the Examiner's combination of Erickson, Germscheid and Wu is improper. The Examiner has failed to respond to this argument in the Response to Arguments.

Thus, for at least the reasons above, the rejection of claim 18 is not supported by the prior art and removal thereof is respectfully requested. Similar remarks also apply to claims 39 and 51.

Regarding both § 103 rejections, Applicants also assert that numerous ones of the dependent claims recite further distinctions over the cited art. However, since the rejection has been shown to be unsupported for the independent claims, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicants submit the application is in condition for allowance, and prompt notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5181-47300/RCK.

Respectfully submitted,

/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: December 14, 2006